CS 6501 Week 14: Advanced Lattice-Based Primitives

So far, we have seen how to leverage lattice homomorphisms to get homomorphic encryption and homomorphic signatures. This week, we will continue and look at another primitive that makes use of homomorphism: attribute-based encryption

Attribute-based encryption: Generalization of identity-based encryption:

- Ciphertexts are associated with an attribute x and a message m
- Secret keys are associated with functions f

} decryption recovers m if $f(x) = 1$ and otherwise, ciphertexts are semantically secure

- Useful notion for enforcing access control (e.g., attribute might be "CONFIDENTIAL" and "TOP-SECRET") and decryption key corresponds to access level

Schema:

$\text{Setup}(1^\lambda) \to (\text{mpk}, \text{msk})$

$\text{KeyGen}(\text{msk}, C) \to \text{sk}_C$

$\text{Encrypt}(\text{mpk}, x, m) \to \text{ct}$

$\text{Decrypt}(\text{sk}_f, \text{ct}) \to m / \perp$

Correctness: for any attribute x and circuit C where $C(x) = 1$,

$(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda)$

$\text{sk}_C \leftarrow \text{KeyGen}(\text{msk}, C)$ $\rightsquigarrow$ $\Pr[\text{Decrypt}(\text{sk}_C, \text{ct}) = m] = 1$

$\text{ct} \leftarrow \text{Encrypt}(\text{mpk}, x, m)$

Semantic Security:

adversary | challenger

$b \in \{0,1\}$

$(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda)$

$\xleftarrow{\text{mpk}}$

$\xrightarrow{C}$

$\xleftarrow{\text{sk}_C \leftarrow \text{KeyGen}(\text{msk}, C)}$

$\xrightarrow{x, m_0, m_1}$

$\xleftarrow{\text{ct} \leftarrow \text{Encrypt}(\text{mpk}, x, m_b)}$

$\xrightarrow{C}$

$\xleftarrow{\text{sk}_C \leftarrow \text{KeyGen}(\text{msk}, C)}$

$b' \in \{0,1\}$

Requirement: $C(x) = 0$ for all circuits queried by the adversary

Selective Security: Adversary commits to the challenge attribute x^* at the beginning of the security game

↳ Selective security implies adaptive security via technique called "complexity leveraging" [reduction guesses the challenge attribute at the beginning of the game] — this incurs a <u>subexponential</u> loss in the security reduction so will require a <u>subexponential</u> hardness assumption:

$$\text{AdaptiveAdv}[B] \leq \frac{1}{2^\ell} \text{SelectiveAdv}[A]$$

where ℓ is the attribute length

Starting point: dual version of Regev's encryption (interchange ciphertexts with secret key): $\left[\begin{array}{l}\text{we can also sample } (A, td_A) \leftarrow \mathsf{TrapGen}(1^\lambda) \\ u \xleftarrow{R} \mathbb{Z}_q^n \text{ and } r \leftarrow f_A^{-1}(td_A, u)\end{array}\right]$

- Setup(1^λ): Sample $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, $r \xleftarrow{R} \{0,1\}^m$ and compute $u \leftarrow Ar \in \mathbb{Z}_q^n$
  Output $pk = (A, u)$ and $sk = r$
- Encrypt(pk, μ): Sample $s \xleftarrow{R} \mathbb{Z}_q^n$, $e \leftarrow \chi^m$, $e' \leftarrow \chi$ and output $ct = (s^T A + e^T, s^T u + e' + \mu \cdot \lfloor \frac{q}{2} \rfloor)$
- Decrypt(sk, ct): Output $\lfloor ct_1 - \langle ct_0, r \rangle \rceil_2$

Correctness:
$$ct_1 - \langle ct_0, r\rangle = s^T u + e' + \mu \cdot \lfloor \tfrac{q}{2} \rfloor - s^T A r - e^T r$$
$$= s^T A r + e' + \mu \cdot \lfloor \tfrac{q}{2} \rfloor - s^T A r - e^T r = \mu \cdot \lfloor \tfrac{q}{2} \rfloor + e' - e^T r$$
Correct as long as $|e' - e^T r| < \frac{q}{4}$

Security: By LHL, public key statistically indistinguishable from sampling $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, $u \xleftarrow{R} \mathbb{Z}_q^n$
Then,
$$(s^T A + e^T, s^T u + e' + \mu \cdot \lfloor \tfrac{q}{2} \rfloor) \overset{c}{\approx} (r, r')$$
where $r \xleftarrow{R} \mathbb{Z}_q^m$, $r' \xleftarrow{R} \mathbb{Z}_q$ by LWE.

Comparison with standard (i.e., primal) Regev:

	primal Regev	dual Regev
public key:	sample $s \xleftarrow{R} \mathbb{Z}_q^n$, $e \leftarrow \chi^m$ A, $s^T A + e^T$	sample $r \xleftarrow{R} \{0,1\}^m$ A, Ar
secret key:	s	r
ciphertext:	sample $r \xleftarrow{R} \{0,1\}^m$ $(Ar, s^T A r + e^T r + \mu \cdot \lfloor \frac{q}{2} \rfloor)$	sample $s \xleftarrow{R} \mathbb{Z}_q^n$, $e \leftarrow \chi^m$, $e' \leftarrow \chi$ $(s^T A + e^T, s^T A r + e' + m \cdot \lfloor \frac{q}{2} \rfloor)$

Trapdoor extension:
1. Suppose we have a gadget trapdoor $R \in \mathbb{Z}_q^{m \times m}$ for $A \in \mathbb{Z}_q^{n \times m}$ (i.e., $AR = G$).
   Then, $\left[\frac{R}{0}\right]$ is a trapdoor for any extension $[A \mid A_1]$ of A: $[A \mid A_1]\left[\frac{R}{0}\right] = AR = G$.
2. For a matrix $A \in \mathbb{Z}_q^{n \times m}$ and $U = AR + G \in \mathbb{Z}_q^{n \times m}$, then $\left[\frac{-R}{I}\right]$ is a trapdoor for $[A \mid U]$:
   $$[A \mid U]\left[\tfrac{-R}{I}\right] = -AR + U = -AR + AR + G = G$$

Useful observation: Two possible trapdoors for a lattice $[A \mid A_1]$: either know a trapdoor for A or
know a short R such that $A_1 = AR + G$

This is a useful tool in many security proofs relying on the "puncturing" technique
- Real scheme will use the trapdoor for A
- Reduction (simulation) will set up parameters so it knows R such that $A_1 = AR + G$
  - ↳ Since reduction likely will reduce to LWE (and no trapdoor is provided!)

We will write SampleLeft(A, A_1, td_A, v, β) to denote an algorithm that sample a pre-image u such that
$$[A \mid A_1] u = v \text{ and } \|u\| \leq \beta$$
We will write SampleRight(A, B, R, v, β) to denote an algorithm that samples a pre-image u such that
$$[A \mid B] u = v \text{ and } \|u\| \leq \beta$$
provided that $B = AR + G$. In both cases, the allowable value of β depends on the quality of the trapdoor (td_A or R)

We start with an <u>abstraction</u> for the homomorphic operations we have examined so far.

<u>Matrix embeddings</u>: Let $A_1, \ldots, A_\ell \in \mathbb{Z}_q^{n \times m}$. Take $x \in \{0,1\}^\ell$. We can "encode" x as follows:

$$\begin{aligned} v_1 &= s^T (A_1 + x_1 G) + e_1^T \\ &\;\vdots \\ v_\ell &= s^T (A_\ell + x_\ell G) + e_\ell^T \end{aligned}$$

<u>Addition</u>: Given $v_i = s^T(A_i + x_i G) + e_i^T$, $v_j = s^T(A_j + x_j G) + e_j^T$

$$\Rightarrow \underbrace{v_i + v_j}_{v_+} = s^T\big(\underbrace{(A_i + A_j)}_{A_+} + (x_i + x_j) G\big) + \underbrace{e_i^T + e_j^T}_{e_+}$$

<u>Multiplication</u>: Given $v_i = s^T(A_i + x_i G) + e_i^T$, $v_j = s^T(A_j + x_j G) + e_j^T$

$$\begin{aligned} \Rightarrow \underbrace{x_j v_i - v_j G^{-1}(A_i)}_{v_\times} &= s^T(x_j A_i + x_i x_j G) + x_j e_i^T - s^T(A_j G^{-1}(A_i) + x_j A_i) + e_j^T G^{-1}(A_i) \\ &= s^T\big(-A_j G^{-1}(A_i) + x_i x_j \cdot G\big) + x_j e_i^T + e_j^T G^{-1}(A_i) \end{aligned}$$

Using these elementary operations, we can define functions

$$\mathrm{EvalPK}(C, A_1, \ldots, A_\ell) \to A_C$$

$$\mathrm{EvalCT}(C, A_1, \ldots, A_\ell, v_1, \ldots, v_\ell, x_1, \ldots, x_\ell) \to v_C$$

such that: for any collection of matrices $A_1, \ldots, A_\ell \in \mathbb{Z}_q^{n \times m}$, if

$$v_i = s^T(A_i + x_i G) + e_i^T \quad \text{for all } i \in [\ell],$$

then if we take

$$A_C \leftarrow \mathrm{EvalPK}(C, A_1, \ldots, A_\ell)$$

$$v_C \leftarrow \mathrm{EvalCT}(C, A_1, \ldots, A_\ell, v_1, \ldots, v_\ell, x_1, \ldots, x_\ell),$$

it follows that

$$v_C = s^T(A_C + C(x) \cdot G) + e_C^T$$

Next, if $A_i = AR_i - x_i G$, then observe that

- $A_i + A_j = A(R_i + R_j) - (x_i + x_j) \cdot G$
- $\begin{aligned}[t] -A_j G^{-1}(A_i) &= -AR_j G^{-1}(A_i) + x_j A_i \\ &= -AR_j G^{-1}(A_i) + x_j A R_i - x_i x_j G \\ &= A\underbrace{(-R_j G^{-1}(A_i) + x_j R_i)}_{R_\times} - x_i x_j G \end{aligned}$

if $A_C \leftarrow \mathrm{EvalPK}(C, A_1, \ldots, A_\ell)$,

then $A_C = AR_C - C(x) \cdot G$

function of $C, A, R_1, \ldots, R_\ell$, and x

ABE from lattices: - use matrix encoding to encode attributes:

$$\left.\begin{aligned} v_1 &= s^T(A_1 + x_1 G) + e_1^T \\ &\vdots \\ v_\ell &= s^T(A_\ell + x_\ell G) + e_\ell^T \end{aligned}\right\}$$ enables computation of $s^T(A_C + C(x)\cdot G) + e_C^T$ $\Rightarrow$ if $C(x) = 0$, this becomes $\underbrace{s^T A_C + e_C^T}$

we will use the convention that $C(x) = 0 \Rightarrow$ can decrypt

looks like part of a dual Regev encryption (with public key A_C)

- encrypt a message μ with respect to vector u (e.g., dual Regev style)

$$s^T u + e' + \mu \cdot \lfloor \tfrac{q}{2} \rfloor$$

- ciphertext is essentially dual Regev encryption with respect to A_C and u; to decrypt, we need to give out a short vector r_C such that $u = A_C r_C$ — seems challenging unless we have a trapdoor for A_C
- will use basis extension to make this easier: instead of encrypting to A_C, we instead encrypt with respect to $[A \mid A_C]$ and let master secret key be trapdoor for A (which can be used to generate trapdoors for $[A \mid A_C]$ — these will be the ABE decryption keys

Setup(1^λ): Sample $(A, td_A) \leftarrow \text{TrapGen}(1^\lambda)$

$A_1, \ldots, A_\ell \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ mpk: $(A, A_1, \ldots, A_\ell, u)$

$u \xleftarrow{R} \mathbb{Z}_q^n$ msk: td_A

Encrypt(mpk, x, μ): Sample $s \xleftarrow{R} \mathbb{Z}_q^n$, $e, e_1, \ldots, e_\ell \leftarrow \chi^m$, $e' \leftarrow \chi$ and compute

$$\begin{aligned} v &= s^T A + e^T \\ v_1 &= s^T(A_1 + x_1 G) + e_1^T \\ &\vdots \\ v_\ell &= s^T(A_\ell + x_\ell G) + e_\ell^T \\ v' &= s^T u + e' + \mu \cdot \lfloor \tfrac{q}{2} \rfloor \end{aligned}$$

ct: $(v, v_1, \ldots, v_\ell, v', x)$

KeyGen(msk, C): $A_C \leftarrow \text{EvalPK}(C, A_1, \ldots, A_\ell)$

output $r_C \leftarrow \text{SampleLeft}(A, A_C, td_A, u, \beta)$ [β is some bound chosen to satisfy correctness and security]

[in particular $[A \mid A_C] r_C = u$

Decrypt(sk_C, ct): if $C(x) = 1$, output $\perp$

$v_C^T \leftarrow \text{EvalCT}(C, A_1, \ldots, A_\ell, v_1, \ldots, v_\ell, x_1, \ldots, x_\ell)$

output $\lfloor v' - [v^T \mid v_C^T] r_C \rceil_2$

Correctness: Take any $x \in \{0,1\}^\ell$ and circuit $C: \{0,1\}^\ell \to \{0,1\}$ where $C(x) = 0$. Consider decrypting a valid ciphertext with attribute x and message μ. Then,

$$\begin{aligned} v &= s^T A + e^T \\ v_C &= s^T(A_C + C(x)\cdot G) + e_C^T = s^T A_C + e_C^T \\ \Rightarrow [v^T \mid v_C^T] r_C &= s^T [A \mid A_C] r_C + [e^T \mid e_C^T] r_C \\ &= s^T u + [e^T \mid e_C^T] r_C \\ \Rightarrow v' - [v^T \mid v_C^T] r_C &= s^T u + e' + \mu \cdot \lfloor \tfrac{q}{2} \rfloor - s^T u - [e^T \mid e_C^T] r_C \\ &= \mu \cdot \lfloor \tfrac{q}{2} \rfloor + \underbrace{e' - [e^T \mid e_C^T] r_C}_{\tilde{e}} \end{aligned}$$

size grows with $m^{O(d)}$ where d is the depth of the computation

Correct as long as $|\tilde{e}| < \frac{q}{4}$ will be small since e', e^T, e_C^T are small, as is r_C

bounded by β (quality of trapdoor)

<u>Security</u>: Give a reduction to LWE. High level idea:

- Use LWE to argue that ciphertext components are uniformly random:

$$\left.\begin{array}{l} s^T A + e^T \\ s^T(A_1 + x_1^* G) + e_1^T \\ \vdots \\ s^T(A_\ell + x_\ell^* G) + e_\ell^T \\ s^T u + e' + \mu \cdot \lfloor \frac{q}{2} \rfloor \end{array}\right\}$$

observe: $A, A_1, \ldots, A_\ell, u$ are random so

$$s^T[A \mid A_1 \mid \cdots \mid A_\ell \mid u] + [e^T, e_1^T \mid \cdots \mid e_\ell^T \mid e']$$

will appear indistinguishable from uniform under LWE

problem: how do we simulate the decryption keys (we do not know trapdoor for A)

- Proof technique will rely on the "puncturing" technique that will allow us to generate keys for all admissible functions f where $f(x^*) = 0$ [x^* is the challenge attribute] and does <u>not</u> require trapdoor for A
- Will consider <u>selective security</u> game where $\mathcal{A}$ chooses its attribute in advance (implies adaptive security via complexity leveraging – though relies on subexponential hardness assumption) – major open problem is to obtain adaptive security <u>without</u> complexity leveraging / subexponential hardness

We use a hybrid argument:

<u>Hyb_0</u>: Real game

<u>Hyb_1</u>: After adversary commits to x^*, we set public parameters as follows:

$$\left.\begin{array}{l} A \xleftarrow{R} \mathbb{Z}_q^{n \times m} \qquad R_1, \ldots, R_\ell \xleftarrow{R} \{\pm 1\}^{m \times m} \\ A_1 \leftarrow AR_1 - x_1^* G \in \mathbb{Z}_q^{n \times m} \\ \vdots \\ A_\ell \leftarrow AR_\ell - x_\ell^* G \in \mathbb{Z}_q^{n \times m} \\ u \xleftarrow{R} \mathbb{Z}_q^n \end{array}\right\} \begin{array}{l} \text{mpk} = (A, A_1, \ldots, A_\ell, u) \\ \text{msk} = T_A \end{array}$$

For challenge ciphertext, compute

$$[v \mid v_1 \mid \cdots \mid v_\ell] = s^T A[I \mid R_1 \mid \cdots \mid R_\ell] + e^T[I \mid R_1 \mid \cdots \mid R_\ell]$$

$$v' = s^T u + e' + \mu \cdot \lfloor \tfrac{q}{2} \rfloor$$

<u>Hyb_2</u>: Switch challenge ciphertext to uniformly random vectors:

$$v, v_1, \ldots, v_\ell \leftarrow \mathbb{Z}_q^m, \quad v' \xleftarrow{R} \mathbb{Z}_q$$

$$ct = (v, v_1, \ldots, v_\ell, x^*)$$

Hyb_0 and Hyb_1 are statistically indistinguishable by LHL. Namely, $(A, AR_1, \ldots, AR_\ell) \overset{s}{\approx} (A, U_1, \ldots, U_\ell)$ if $A, U_1, \ldots, U_\ell \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and $R_1, \ldots, R_\ell \xleftarrow{R} \{\pm 1\}^{m \times m}$ and $m \geq 3n \log q$.

↳ Strictly speaking, we require a generalization of the LHL that says that $(A, AR, e^T R) \overset{s}{\approx} (A, U, e^T R)$

Hyb_1 and Hyb_2 are computationally indistinguishable by LWE. Suppose there is an adversary $\mathcal{A}$ that can distinguish Hyb_1 and Hyb_2. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ for LWE:

1. Algorithm $\mathcal{B}$ receives an LWE challenge $([A|u],[b|b']) \mathbb{Z}_q^{n\times(m+1)} \times \mathbb{Z}_q^{m+1}$ where $\begin{matrix} b^T = s^TA+e^T \\ b' = s^Tu + e' \end{matrix}$ or $\begin{matrix} b \xleftarrow{R} \mathbb{Z}_q^m \\ b' \xleftarrow{R} \mathbb{Z}_q \end{matrix}$.
2. Let x^* be the attribute $\mathcal{A}$ chooses for the semantic security game.
3. Algorithm $\mathcal{B}$ samples $R_1,\dots,R_\ell \xleftarrow{R} \{\pm 1\}^{m\times m}$ and sets $A_i \leftarrow AR_i - x_i^* G$ and sets the mpk as $(A, A_1, \dots, A_\ell, u)$ and gives mpk to $\mathcal{A}$.
4. Suppose $\mathcal{A}$ makes a key-generation query on a circuit C. It must be the case that $C(x^*) = 1$. This means that

$$\begin{aligned} A_C &= \mathrm{EvalPK}(C, A_1, \dots, A_\ell) \\ &= \mathrm{EvalPK}(C, AR_1 - x_1^* G, \dots, AR_\ell - x_\ell^* G) \\ &= AR_C + C(x^*)\cdot G \\ &= AR_C + G \end{aligned}$$

This will allow the reduction to sample keys whenever $C(x^*) = 1$, but not when $C(x^*) = 0$.
[known as the "puncturing" technique — we have a trapdoor that works sometimes]

   By design, R_C is small. To simulate a key, algorithm $\mathcal{B}$ needs to compute a short r_C such that $[A|A_C]r_C = u$. This is possible since $\mathcal{B}$ knows R_C such that $A_C = AR_C + G$ so $\mathcal{B}$ computes $r_C \leftarrow \mathrm{SampleRight}(A, A_C, R_C, u, \beta)$, which is indistinguishable from a real key (output by the actual KeyGen algorithm)
5. For the challenge ciphertext, set

$$v = b \quad\text{and}\quad v_i = b^T R_i \ \text{ for } i \in [\ell]$$
$$v' = b' + \mu \cdot \lfloor \tfrac{q}{2} \rfloor$$

   and output $ct = (v, v_1, \dots, v_\ell, v')$.

<u>Two possibilities</u>: – Suppose $b^T = s^TA + e^T$ and $b' = s^Tu + e'$. Then,

$$\begin{aligned} v_i = (s^TA + e^T)R_i &= s^TAR_i + e^TR_i \\ &= s^T(A_i + x_i^* G) + e^TR_i \end{aligned}$$

Thus, ciphertexts distributed exactly as in Hyb_1.

– Suppose b^T and b' are uniformly random. Then, by LHL, all of the v_i are uniform over $\mathbb{Z}_q^m$ and the ciphertext is distributed according to the specification in Hyb_2.

Thus, assuming LWE, Hyb_1 and Hyb_2 is computationally indistinguishable.